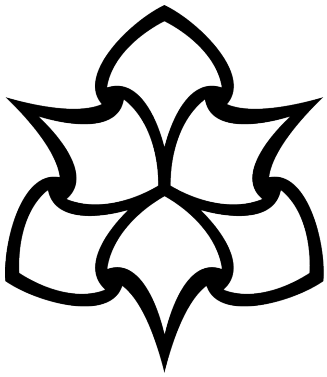

Performance Analysis of Data Fragmentation Techniques on a Cloud Server

Nelson Santos[1],Salvatore Lentini[1*], Enrico Grosso[2], Bogdan Ghita[3], Giovanni Masala[1*]

[1]Big Data Group, University of Plymouth, Drake Circus, PL4 8AA, Plymouth, United Kingdom

nelson.santos@students.plymouth.ac.uk; Salvatore.lentini@postgrad.plymouth.ac.uk; Giovanni.masala@plymouth.ac.uk;

[2]Centre for Security, Communications and Network Research, University of Plymouth, Drake Circus, PL4 8AA, Plymouth, United Kingdom

Bogdan.ghita@plymouth.ac.uk

[3]Computer Vision Laboratory, University of Sassari, Viale Mancini, 5, 07100, Sassari, Italy

grosso@uniss.it

Abstract: The advancements in virtualization and distributed computing have allowed the cloud paradigm to become very popular among users and resources. It allows companies to save costs on infrastructure and maintenance and to focus on the development of products. However, this fast-growing paradigm has brought along some concerns from users, such as the integrity and security of the data, particularly in environments where users rely entirely on providers to secure their data. This paper explores different techniques to fragment data on the cloud and prevent direct unauthorized access to the data. It explores their performance on a cloud instance, where the total time to perform the operation, including the upload and download of the data, is considered. Results from this experiment indicate that fragmentation algorithms show better performance compared to encryption. Moreover, when combining encryption with fragmentation, there is an increase in the security, with the trade-off of the performance.

Keywords: Cloud Security, Data Fragmentation, Data Security, Privacy in Cloud Computing

Reference to this paper should be made as follows: Author(s) (2006) 'paper title ', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. X, No. Y4, pp.000–000.

1 Introduction

Cloud computing has grown in such a way that can be considered one of the most promising IT paradigms, in which most applications are now hosted as services on the Internet. Such services can be divided into three main categories: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). NIST (2011) defines cloud computing as a model that allows access to a pool of resources such as networks, storage or applications that are provisioned with minimal effort from the provider. In this scenario, virtualization and distributed computing are the cornerstones. This allows the customers to reduce the cost of the storage and computing clusters, as well deviate from the burden of maintaining the infrastructure and shift all the focus towards the development of applications (Bahrami & Singhal, 2015).

Although cloud computing brought many benefits, it also generated a number of challenges. Among them, the protection of the data being stored in the

cloud and the privacy of the users are the most significant ones. Surveys conducted by the Intel IT Center (2012) and the Cloud Security Alliance (2013), indicated that the top three cloud security concerns are the inability to measure the provider's security services, the lack of control over data and the confidence in the capabilities of the provider. In addition, the data is handled by the provider, which also oversees its safekeeping. According to the Cloud Security Alliance, (2010), Kumar & Raj (2018) and Hegarty & Haggerty (2015) the cloud provider often does not disclose internal procedures on storing and safekeeping the data to the user. Furthermore, many of the organizations that provide cloud services use data mining techniques to extract information from the clients and utilize or sell such information, often for advertising purposes, as described by Chow, et al. (2009) and Dev, et al. (2012). Such behavior exposes users to attackers with unauthorized access to the cloud (Dev, et al., 2012). Encryption schemes often satisfy the data privacy problem, however, they bring forward performance issues, such as the complexity and computationally expensive nature of the encryption algorithms (Bahrami & Singhal, 2015 and Bahrami & Singhal, 2016). As a result, researchers shifted their focus on alternative measures to protect the privacy of users. This paper explores the use of data fragmentation in the cloud, by analyzing the performance of different fragmentation algorithms on a cloud instance, hosted in the Amazon Web Services (AWS), from (Amazon, 2018). It will start by analyzing the state of the art in fragmentation algorithms, followed by an explanation of the different methods. Each mechanism will be thus evaluated, and results analyzed and compared with AES (Federal Information Processing Standards, 2001), a common encryption algorithm. Furthermore, the combination of AES and Random Pattern fragmentation is analyzed, showing that this approach allows for the highest level of security among all the tested methods. The comparison of the methods gives a better understanding of each mechanism, along with their benefits and drawbacks.

2 State of the art

This section will highlight the current state of the art with regards to research performed in the data privacy on the cloud. It will investigate the use of data anonymization, and data fragmentation.

The research community attempted to solve the privacy on the cloud with various approaches, some of which include encryption, and data anonymization. As an example, Goswami and Madan (2017) studied various well-known anonymization methods for their advantages and disadvantages. Barak, et al. (2016), applied semantic labelling to achieve anonymization by replacing location coordinates with semantic categories. Ghinita, et al. (2007) attempted to solve K-anonymity and l-diversity problems by mapping multidimensional identifiers on a single dimension. In Jang (2017), the author proposes a method based on deep anonymization for big data, to aid in the reduction of information loss. Furthermore, Gkoulalas-Divanis and Loukides (2011), address the issue of information loss by using a method based on clustering. However, this method may allow identification of an individual based on their sensitive information. Jesu, et al. (2017) also proposed a method based on clustering, using the Hadoop Distributed File System. Al-Zobbi, et al. (2015), proposed a novel anonymization framework that takes a bottom-up approach on the data and applies sensitivity on the anonymization process instead of generalising equivalent records. This approach is suitable for big data environments and is compatible with the MapReduce model. Furthermore, Canbary and Sagiroglu (2017), proposed the use of *spark* and MapReduce to anonymize streaming big data.

Some works where data fragmentation has been applied as a mean to provide privacy include Kapusta & Memmi (2015), who attempted to avoid encryption by separating the data into distinct groups, each with a distinct level of security, based on the sensitivity of the data being stored. However, when faced with large datasets, the running time of their algorithm increased due to the number of clusters formed. Hegarty & Haggerty (2015) presented a system of extrusion detection of files that are maliciously uploaded or downloaded in the cloud. Dev, et al. (2012), approached the problem by categorizing and fragmenting the data, followed by storing the data on different providers. Nevertheless, the constant access to the data hinders the performance of the algorithm. Authors in Memmi, et al. (2016) propose more complex solutions, which include the use of GPUs to

incorporate fragmentation, encryption and dispersion. Ciriani, et al (2010), also addressed the data privacy issue by combining encryption with fragmentation, by modelling the sensitivity and the data after encryption, followed by using fragmentation to break the association among attributes.

To improve the management of data within the cloud, researchers investigated the use of a database to combine with fragmentation. For instance, Alsirhani, et al. (2017), proposed a combination of encryption algorithms and distributed a database across different cloud providers, based on the encryption security level. Aggarwal, et al. (2005), explored different techniques to decompose data and optimize queries in a distributed database. Masala et al. (2018), proposed an approach of storing fragmented data with a MongoDB database. Furthermore, Santos et al (2018) investigated the use of random pattern fragmentation to chunk data and save on a NoSQL database. El Mrabti, et al. (2017), investigated the possibility of applying data fragmentation on Android devices, to allow different policy strategies for applications that need to access data from the device.

This work will focus on the scenario where the data is stored in a single cloud provider, considering that this is the least recommended approach, given all the data will be present in the same location, where an attacker inside the cloud could access. Moreover, users may find many occurrences such as the cloud provider running out of business, or having data backed up on the same provider, as it will void the intended security measures because the complete data will be accessible through the backups.

Nevertheless, current work can be extended to work with more data types, including but not limited to general pictures and medical images, or it can be used on less efficient devices such as smartphones. Other different scenarios can also be considered when applying these techniques. For instance, the analysed techniques use multiple SSH sessions to send the split files to the cloud provider. Considering a scenario with different providers, a connection can be opened with each provider and the split files can be sent concurrently. It is important to note that cloud providers have different speeds and performance can be affected by the presence of additional and uncontrolled variables; these problems, however, go beyond the scope of this paper. Nowadays, business and companies tend to use the cloud to back up their data. The methods can be applied on such backups to protect them from unauthorized access within the cloud. The data anonymization techniques described earlier can also be used to add an extra layer of security on the data.

3 Fragmentation Algorithms

Before explaining the pattern fragmentation algorithms, the permutation approach must be detailed. It was introduced by Bahrami & Singhal (2015), where the authors proposed a light-weight method for mobile clients to store data on one or multiple clouds using a pseudo random permutation based on chaos systems (Gharajedagh, 2011). This is less computationally expensive, compared to operations such as secret key or public-key encryptions, but provides a good balance between security and efficiency, especially for devices with limited resources such as mobile phones. The author's proposal is optimized for JPEG images and, when compared to encryption algorithms such as AES or JPEG encoders, it proved more efficient than the counterparts, whilst to an extent, protecting the user data privacy.

The algorithm reads binary files rather than specific formats, and it is divided into two stages to split a file and recombine it:

- **Disassembling (fragmentation):** the original file is split into multiple chunks and the chunks are inserted into binary files, (split files), based on a pattern using the chaos system (Bahrami & Singhal, 2015). A pattern can be defined as a key for the user or can be randomly selected. Users are also able to define different patterns to provide a different strategy for the distribution. The output is then stored into the cloud.
- **Assembly (Recovery):** The split files are recombined to reorganize the original file. The scrambled files are downloaded from the cloud and the chaos system random arrays are reordered based on the pattern that fragmented them initially.

In this implementation of the method, the user is also able to configure the application to set the:

- Number of split files

- The size of the chunks
- The User account in the cloud to upload/download the files.

3.1 Predefined Pattern Fragmentation

In the predefined pattern fragmentation (figure 1), the chunks are inserted in a split file with an odd or even index. After splitting the original file, the chunks are stored in the split file according to the index they receive. As a result, only two split files are created and the length of each chunk is calculated. Using this method, the attacker will need knowledge of the length of the chunk to reconstruct the file.

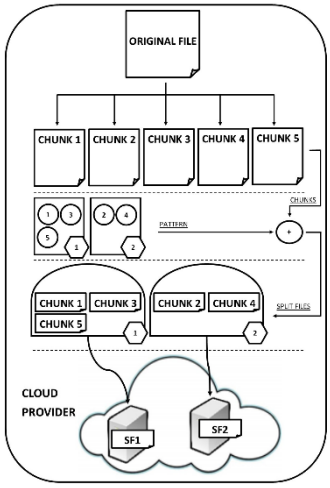

Figure 1 Predefined Pattern Fragmentation method (Fragmentation steps). After splitting the file, the chunks receive an odd or even index. Based on the index given, the chunks are then inserted on a split file.

In the reconstruction stage (figure 2), the split files are downloaded and opened in the same order in which they were created, based on the length of the chunks. The chunks from each split file are stored in a dictionary data structure, where the data is associated with a key. This key contains the pattern list in which the objects are then organized in their original position. The result of this operation is then saved on the client device, which constitutes the reconstructed file.

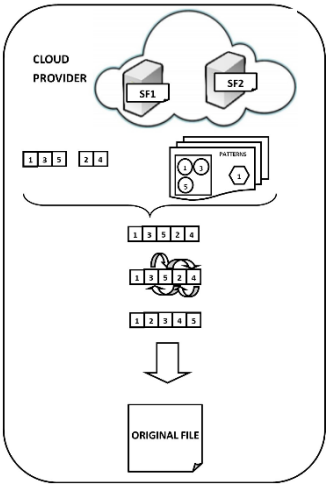

Figure 2 Predefined Pattern Fragmentation method (Recovery steps). The split files are downloaded from the cloud and the chunks are stored in memory as a dictionary data structure. The file is then reconstructed with the keys of the dictionary, which are the indexes assigned to the chunks.

3.2 Random Pattern Fragmentation

In this method, a random function was implemented based on the chaos theory presented in Bahrami & Singhal (2015), i.e., a permutation of a number of N elements, set by the user, is used to calculate the pattern indexes. The original file is divided into N chunks, similar to the other methods, and is then inserted in split files, where the length of each split file is equal to the length in the associated pattern, as demonstrated in figure 3. The highlight of this method is that an attacker will not know the length of each chunk, nor the order in which the chunks are distributed in each split file.

In the original method by Bahrami & Singhal (2015) the use case used was based on images. The header is stored alone on a separate file, with a smaller size, compared to the other split files. It is recommended that this header is not transmitted to the cloud, to hinder attacker from using it to start the reconstruction. In the proposed implementation, padding bytes were added to the header file to mask the length of the file before uploading to the cloud, to hinder attackers from using this file, as they would not understand which is the header, as it is the same size as the other split files.

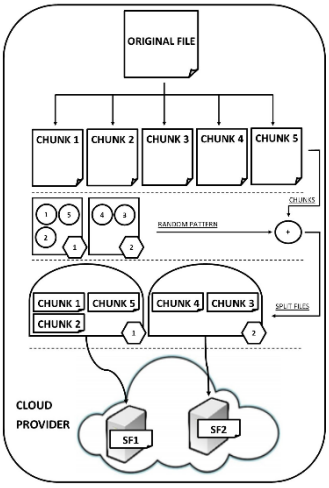

Figure 3 Random Pattern Fragmentation method (Fragmentation steps). The file is split into chunks and those chunks are then inserted into a split file in a random order. In cases where one chunk would have a smaller size than the rest, padding was added to the end of the chunk to create a symmetric size across all chunks.

During the reconstruction phase, the same dictionary based reconstruction described in the predefined pattern fragmentation is used, as shown in Figure 4.

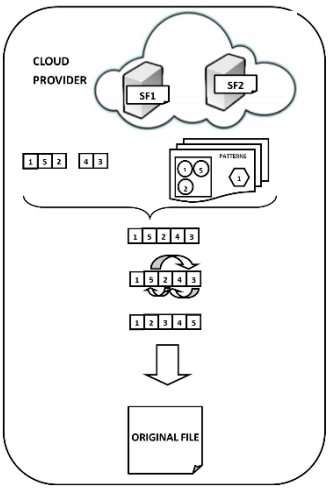

Figure 4 Random Pattern Fragmentation method (Recovery steps). The process is similar to the predefined pattern, with the only difference being that the indexes are in a random order.

3.3 Simple AES Encryption

AES is the most common encryption algorithm used nowadays (Prabhu & Paramesha, 2017). It is defined as a symmetric encryption which uses the same key for both encrypting and decrypting data. Despite the same key being used, it provides a high level of security when encrypting. The algorithm supports block lengths of 128 bits and key sizes of 128,192 and 256 bits in the CBC version. For this experiment, the original file is encoded with AES 256 before being sent to the cloud. Unlike the previous methods, the file is not fragmented. This method was considered in the experiment not only to compare its performance with the other methods, but also to investigate the performance and suitability of a combination of a highly used encryption algorithm and data fragmentation. The same file is then downloaded and decoded as represented in figure 5.

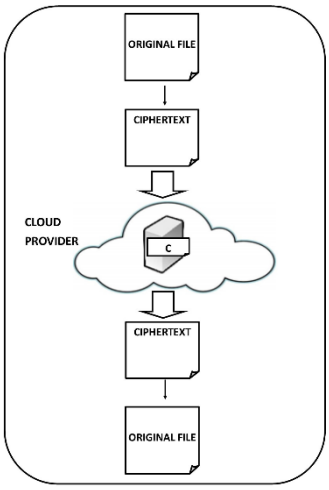

Figure 5 Simple encryption AES 256 (Encoding and decoding steps). The whole original file is encrypted, and it sent to the cloud as a unique file. Vice versa in the decoding phase from the cloud only one single file produces the original file.

3.4 Random Pattern Fragmentation combined with AES 256

This proposed implementation combines the use of random pattern fragmentation with AES for encryption. It has been designed to provide a higher level of security compared to the counterparts, with the burden having encryption (time and computationally expensive). The idea is encrypting the original file with AES 256 CBC and divide the cypher text into chunks. The chunks are arranged using a random pattern before being stored in split files. Each split file is finally sent to the cloud, as shown in figure 6.

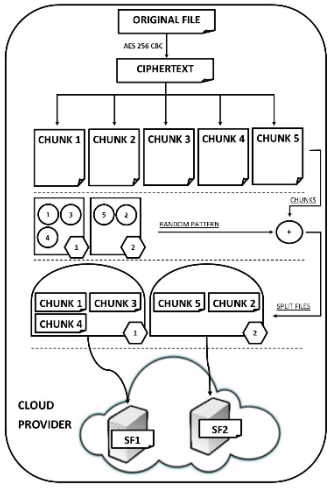

Figure 6 Random Pattern Fragmentation Encryption AES 256 (Fragmentation steps). The original file is encrypted and subdivided in chunks. Each chunk is stored in one of two split files through a random selection. Finally, the split files are sent to the cloud.

When reconstructing the file, as shown in Figure 7, the split files are downloaded and read in sequence, until all the chunks are extracted. The cypher text is recreated using the defined pattern, similarly to the random pattern algorithm explained previously. Finally, the cypher text is decoded with the key and the reconstructed file is stored in the client device.

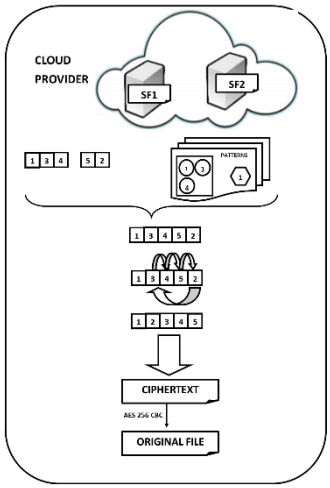

Figure 7 Random Pattern Fragmentation Encryption AES 256 (Recovery steps). The original file is reconstructed similar to the previous implementations and after reconstructing the cypher text, it gets decoded by the key and stored in the client device.

4 Experiment

This section sets the baseline of the conducted experiments. A dataset of four files with different extensions, .jpeg, .docx, .pdf, and .bmp respectively, all with 100KB in size, was used throughout the experiments. The files were uploaded to the program, where the user would be able to set parameters, such as the length of chunks or the number of split files. For this experiment, we used two split files with a chunk length of 1000 bytes. An AWS (Amazon, 2018) micro instance, with the Ubuntu image, was used throughout this experiment to upload and download the test files. The connection between the instance and the client machine was made via SSH. For each of the split files created, an SSH connection was established asynchronously to send the split file. However, this experiment considers the performance of the algorithm independently of external factors, such as the network data rate. The overall time presented includes the fragmentation process, the uploading and downloading, with the reconstruction of the file. The client device used was an Intel Core i7 - 6500U CPU 2.50 GHz, 8 GB of RAM on 64 Bit - Windows 10.

It is important to note that in all the methods described, the chunks are the same size. This is achieved through adding a few bytes of padding when needed. All the results are reported in a file on the user machine.

5 Results

As mentioned in earlier sections, the aim of this paper is to compare the performance of different fragmentation algorithms on a cloud server, to analyse the pros and cons of each algorithm. Before experimenting with the cloud, a local analysis was performed with various chunk sizes, to determine the size that would provide the best performance. It can be seen in figure 8 that bigger chunk sizes present better performance, compared to smaller chunk sizes. This is due to the iterations on the code that directly affect the performance, as bigger chunks lead to less iterations in the loop. Consequently, for the cloud experiment, the chunk size chosen was 1000 bytes.

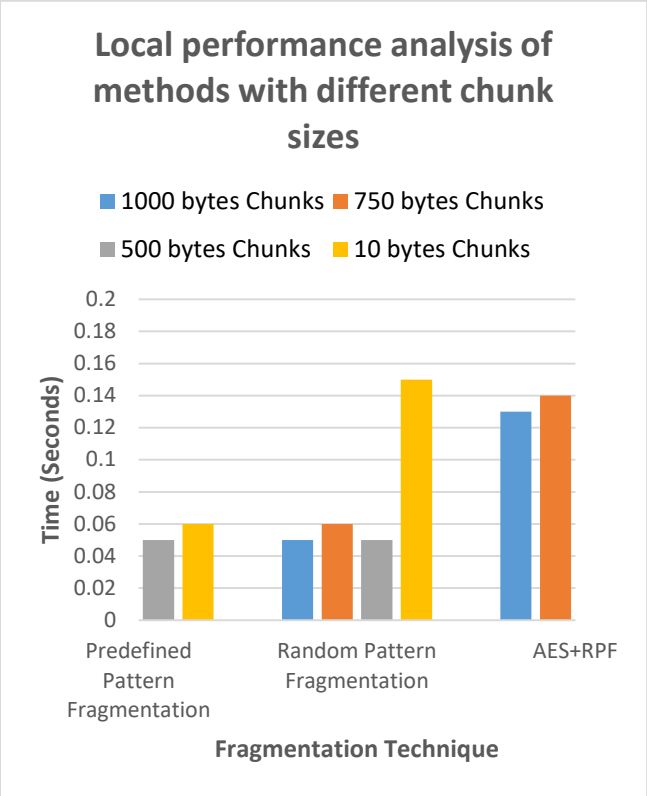

Figure 8 Local Performance analysis with various chunk sizes

During the experiment, we only consider sending all the files to a single instance on a single provider, considering it to be the worst-case scenario, however, this being the most common scenario on the public cloud. For comparison purposes, the time for the same file type to be uploaded and downloaded from the cloud without any techniques, was introduced.

We can see that the difference between the predefined pattern fragmentation (figure 9) and the random pattern fragmentation (Figure 10) can be considered minimal across all the different file types. The random pattern fragmentation proves to be slightly slower than its counterpart, given that the chunks are scrambled in a random order. When compared to the files where no fragmentation was applied, a slight delay is also seen on both graphs, considering the time to apply the fragmentation and the defragmentation. Nevertheless, such trade-off is considered acceptable, considering that applying the techniques will increase the protection of the data.

Regarding the security, the algorithms work with binary data rather than specific formats. This increases the complexity of retrieving the files and provides an additional security layer, as attackers will not be able to discover the pattern in which the chunks are organized.

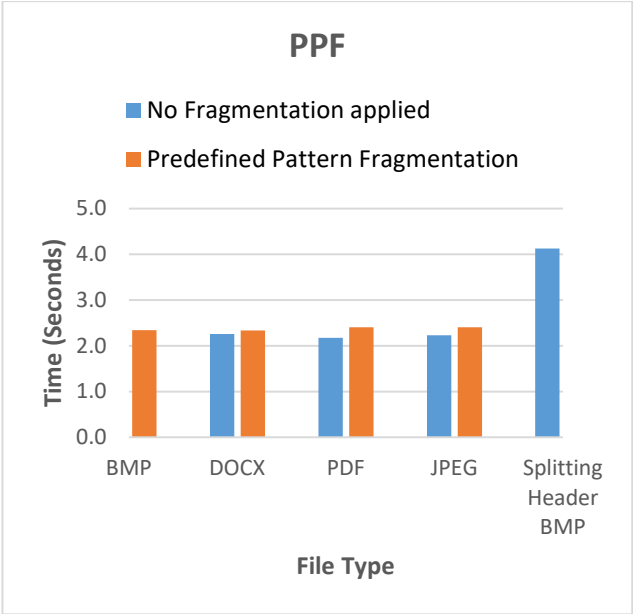

Figure 9 Predefined Pattern fragmentation results with a comparison of the same file without the technique applied

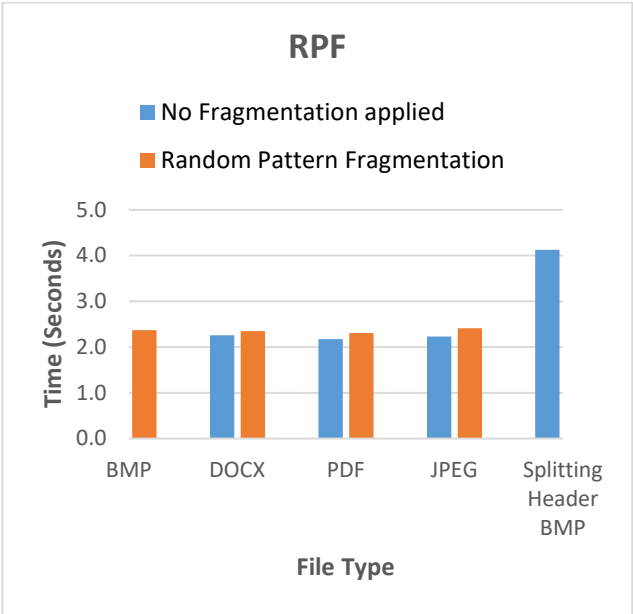

Figure 10 Random pattern fragmentation results with a comparison of the same file without the technique applied

The use of AES encryption to protect data has increased in recent years. It provides a high level of protection on the data by using a key to encrypt and decrypt data. Compared to other methods, AES encryption utilizes many computing resources as the process of encoding and decoding data is time expensive. This can be visualized on Figure 11, where, in some data types, the process takes longer

than 2.5 seconds. As in the previous graphs, the time to split the header for the .bmp file is higher than the other counterparts. However, using the encryption the process is more than 4 seconds, which in computation terms is very high.

Another approach analysed was the combination of the encryption algorithm with the most secure fragmentation algorithm, which is the random pattern fragmentation algorithm to explore how consuming would be to explore the most secure algorithms and provide the highest level available level of security. As it can be seen in Figure 12, this approach is the most time consuming compared to all the others. However, this trade-off allows for the highest level of security on the data, as the chunks are not only scrambled, but also encrypted with a key, making it therefore very difficult to access the data. It is also important to note that this approach would not be suitable in environments where the data needs constant access, as it would consume high amounts of computing resources and time.

Table 1 provides the main properties of each approach summarized. It is notable that the predefined and random pattern fragmentations are good solutions to the data privacy problem, when considering devices with limited resources, such as mobile phones. Where resources allow, combining the random pattern fragmentation with the AES encryption would significantly increase the security of the data, with the performance trade-off.

Table 1 Summary of the properties of each algorithm. It ranks the security, performance and suitability of each method, from low to high.

Method	Sec.	Perform.	Suit.
PPF	Low	High	Mobile Big Data
RPF	Med	High	Mobile Big Data
AES	High	Low	High Security Environments
AES + RPF	High	Low	High Security Environments

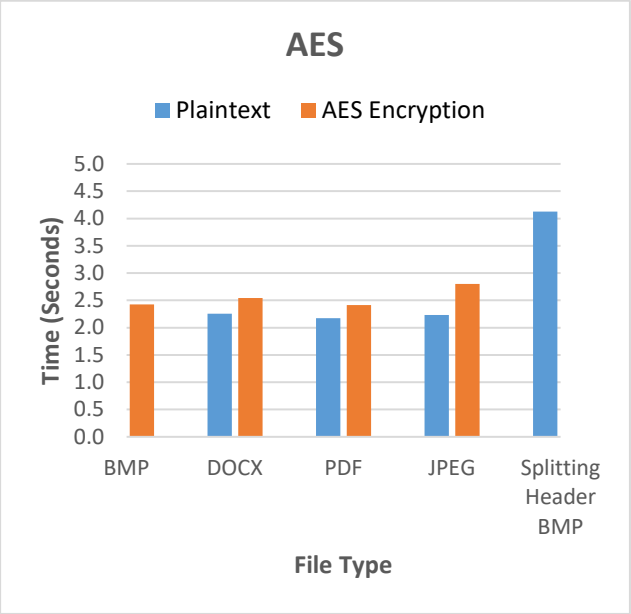

Figure 11 AES encryption results with a comparison of the same file without the technique applied

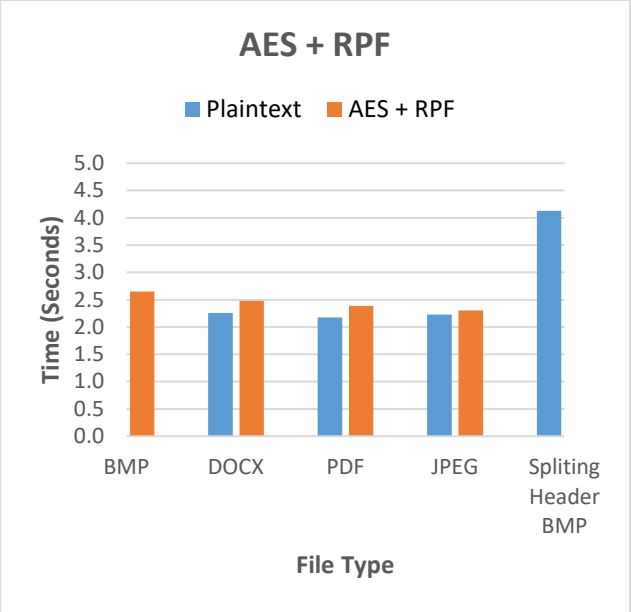

Figure 12 AES Encryption with Random Pattern Fragmentation Results with a comparison of the same file without the technique applied

When comparing the average across the fragmentation algorithms (figure 13), the difference is minimal, meaning that they consume similar resources, apart from the algorithms where encryption is involved, which take on average more than 2.5 seconds. Encryption algorithms also have the highest standard deviation, as there are more processes involved, which includes external factors outside the scope of this experiment. Furthermore, splitting the header and sending to the provider, proves to add a high level of

complexity, as it is the operation that takes more time to complete across all algorithms. Whilst storing this header locally can be considered interesting, it would provide practical issues regarding the management of this header, in scenarios where multiple files are considered, increasing the processing time further.

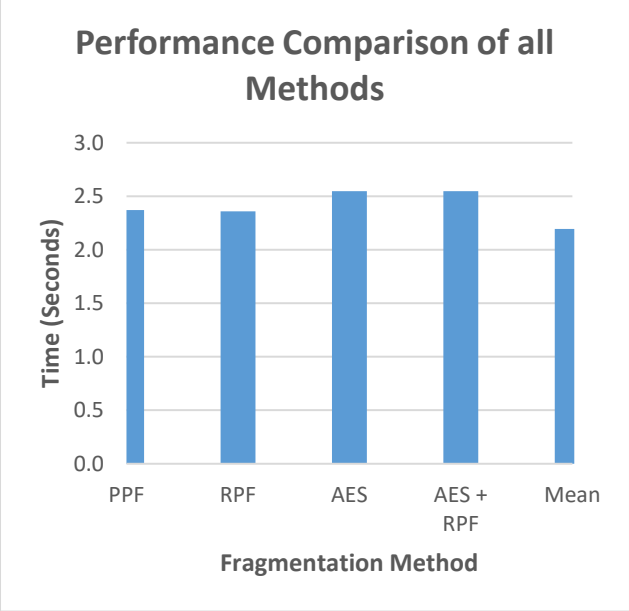

Figure 13 Mean comparison of all methods. The standard deviation of each method is also illustrated on each bar.

6 Conclusion

The aim of this paper is to provide an in-depth performance comparison for a number of methods to secure data in a cloud environment and to ensure the privacy of users from outside attackers, in particular those with access to the cloud provider.

An analysis of time taken to perform each algorithm was performed, while considering different possibilities of securing (based on pattern fragmentations, encryption or both). It was determined that for devices with lower computational abilities, securing the data using pattern fragmentation provides a good level of security without consuming much of the resources. On the other hand, utilizing encryption is recommended on high resource devices, where the extra time would be handled by the higher resources available. On environments of big data, where the privacy and the performance are both priorities, although encryption would favour protection, its resource consumption would affect the overall performance, but utilizing fragmentation to secure the data would be the plausible approach. Some limitations identified with these methods include the continuous access to the data or a multi-user environment, where the techniques are constantly applied probably affecting therefore the performance. Furthermore, the proposed methods do not take into account the management of the data. Therefore, it is advised to store the output of the program into a database, where the data can be managed more easily. Possible future developments would include combining the described techniques with a database to provide a higher level of management of the data, especially in big data environments.

7 References

Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D. and Xu, Y. (2005). Two Can Keep a Secret: A Distributed Architecture for Secure Database Services. In: *CIDR*. [online] Springer. Available at: https://www.microsoft.com/en-us/research/wp-content/uploads/2005/01/storage-cidr.pdf [Accessed 29 Apr. 2018].

Alsirhani, A., Bodorik, P. and Sampali, S. (2017). Improving Database Security in Cloud Computing by Fragmentation of Data. In: *2017 International Conference on Computer and Applications (ICCA)*. Doha: IEEE, pp.43-49.

Al-Zobbi, M., Shahrestani, S. and Ruan, C. (2016). Sensitivity-Based Anonymization of Big Data. In: *41st Conference on Local Computer Networks Workshops (LCN Workshops)*. [online] Dubai: IEEE, pp.58-64. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7856138&isnumber=7855949 [Accessed 29 Apr. 2018].

Amazon Web Services, Inc. (2018). *Amazon Web Services (AWS) - Cloud Computing Services*. [online] Available at: https://aws.amazon.com/ [Accessed 29 Apr. 2018].

Bahrami, M. and Singhal, M. (2015). A Light-Weight Permutation Based Method for Data Privacy in Mobile Cloud Computing. In: *2015 3rd*

IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. [online] San Francisco: IEEE, pp.189-198. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7130886&isnumber=7130853 [Accessed 29 Apr. 2018].

Bahrami, M. and Singhal, M. (2015). The Role of Cloud Computing Architecture in Big Data. In: W. Pedrycz and C. SM, ed., *Information Granularity, Big Data, and Computational Intelligence. Studies in Big Data*, 8th ed. [online] Springer, Cham. Available at: https://link.springer.com/chapter/10.1007%2F978-3-319-08254-7_13 [Accessed 29 Apr. 2018].

Bahrami, M. and Singhal, M. (2016). CloudPDB: A light-weight data privacy schema for cloud-based databases. In: *2016 International Conference on Computing, Networking and Communications (ICNC)*. [online] Kauai: IEEE, pp.1-5. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7440634&isnumber=7440540 [Accessed 29 Apr. 2018].

Barak, O., Cohen, G. and Toch, E. (2016). Anonymizing mobility data using semantic cloaking. *Pervasive and Mobile Computing*, [online] 28, pp.102-112. Available at: https://www.sciencedirect.com/science/article/pii/S1574119215001972 [Accessed 29 Apr. 2018].

Canbay, Y. and Sağıroğlu, S. (2017). Big data anonymization with spark. In: *2017 International Conference on Computer Science and Engineering (UBMK)*. [online] Antalya: IEEE, pp.833-838. Available at: https://ieeexplore.ieee.org/document/8093543/ [Accessed 29 Apr. 2018].

Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R. and Molina, J. (2009). Controlling data in the cloud: outsourcing computation without outsourcing control. In: *Proceedings of the 2009 ACM workshop on Cloud computing security (CCSW '09)*. ACM, pp.58-90.

Ciriani, V., Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S. and Samarati, P. (2010). Combining fragmentation and encryption to protect privacy in data storage. *ACM Transactions on Information and System Security*, [online] 13(3), pp.1-33. Available at: https://dl.acm.org/citation.cfm?id=1805978 [Accessed 29 Apr. 2018].

Cloud Security Alliance (2010). Top Threats to Cloud Computing. [online] Cloud Security Alliance. Available at: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf [Accessed 29 Apr. 2018].

Dev, H., Sen, T., Basak, M. and Ali, M. (2012). An Approach to Protect the Privacy of Cloud Data from Data Mining Based Attacks. In: *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. [online] Salt Lake City: IEEE, pp.1106-1115. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6495915&isnumber=6495777 [Accessed 29 Apr. 2018].

El Mrabti, A., Ammari, N., El kalam, A., Ouahman, A. and De Montfort, M. (2017). Mobile app security by fragmentation "MASF." In: *Proceedings of the Second International Conference on Internet of things and Cloud Computing (ICC '17)*. New York: ACM, p.6.

Federal Information Processing Standards (2001). *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*.

Gharajedaghi, J. (2011). *Systems thinking Systems Thinking: Managing Chaos and Complexity: A Platform for Designing Business Architecture*. Burlington, MA: Elsevier.

Ghinita, G., Karras, P., Kalnis, P. and Mamoulis, N. (2007). Fast data anonymization with low information loss. In: *Proceedings of the 33rd international conference on Very large data bases (VLDB '07)*. Vienna: ACM, pp.758-766.

Gkoulalas-Divanis, A. and Loukides, G. (2011). PCTA: privacy-constrained clustering-based transaction data anonymization. In: *Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society*. New York: ACM.

Goswami, P. and Madan, S. (2017). Privacy preserving data publishing and data anonymization approaches: A review. In: *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, pp.139-142.

Hababeh, I. (n.d.). *A Novel Cloud Computing Data Fragmentation Service Design for Distributed Systems*.

Hegarty, R. and Haggerty, J. (2015). Extrusion detection of illegal files in cloud-based systems. *International Journal of Space-Based and Situated Computing*, 5(3), p.150.

Jang, S. (2017). A study of performance enhancement in big data anonymization. In: *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*. Kuta Bali: IEEE, pp.1-4.

Kapusta, K. and Memmi, G. (2015). Data protection by means of fragmentation in distributed storage systems. In: *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*. Paris: IEEE, pp.1-8.

Kumar, P., Raj, P. and Jelciana, P. (2018). Exploring Data Security Issues and Solutions in Cloud Computing. *Procedia Computer Science*, 125, pp.691-697.

Masala, G., Ruiu, P. and Grosso, E. (2018). Biometric Authentication and Data Security in Cloud Computing. In: K. Daimi, ed., *Computer and Network Security Essentials*. Springer, Charm.

Memmi, G., Kapusta, K. and Qiu, H. (2018). Data protection: Combining fragmentation, encryption, and dispersion. In: *2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)*. Shanghai: IEEE, pp.1-9.

Nayahi, J. and Kavitha, V. (2017). Privacy and utility preserving data clustering for data anonymization and distribution on Hadoop. *Future Generation Computer Systems*, 74, pp.393-408.

NIST (2011). *The NIST Definition of Cloud Computing*. Gaithersburg: National Institute of Standards and Technology.

Prabhu, M. and Paramesha, K. (2018). An Approach for Efficient Utilization of Public Cloud Storage and Securing Data. *International Research Journal of Engineering and Technology (IRJET)*, 4(5), pp.841-844.

Santos, N. and Masala, G. (2018). Big Data Security on Cloud Servers. In: *11th International KES Conference on Intelligent Interactive Multimedia: Systems & Services*. Gold Coast: Springer.